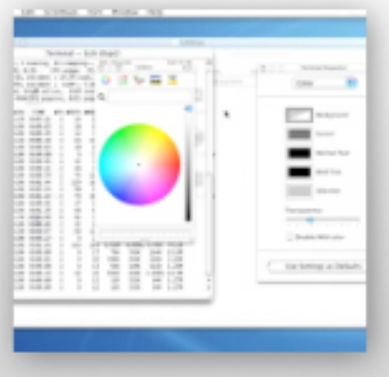
Kernel Programming Interfaces for Extensions

Session 107



Kernel Programming Interfaces for Extensions

Craig Keithley
I/O Technology Evangelist
keithley@apple.com

Kernel Programming Interfaces for Extensions

Simon Patience
Director of Engineering
spatience@apple.com

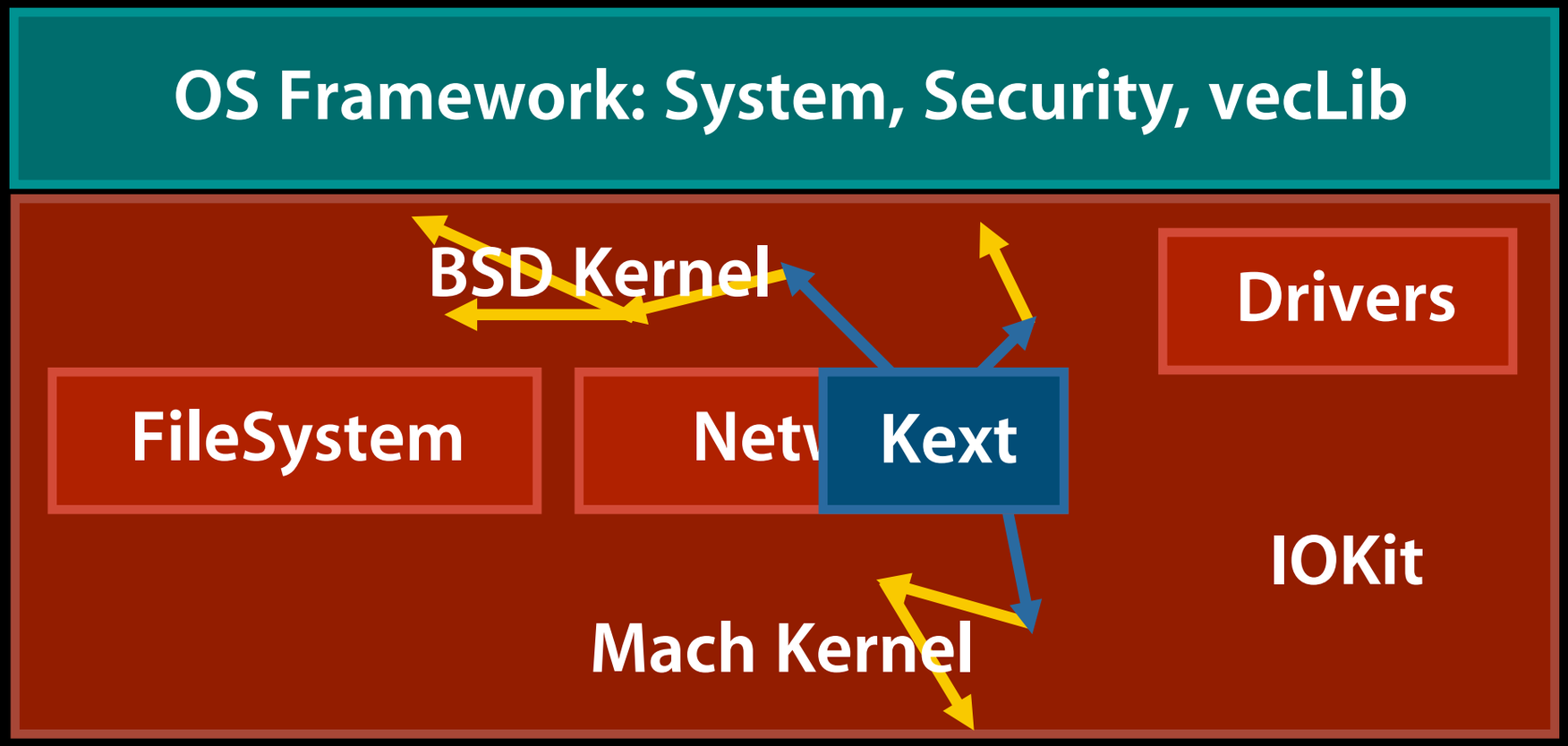
What You'll Learn Today

- Definition, rationale and roadmap of Kernel Interfaces
- Kernel Interface Version Management
- IOKit Interfaces
- General Kernel Interfaces
- File System Interfaces
- Networking Interfaces
- Summary
- Q&A

What Are KPIs?

- Kernel Programming Interfaces
- Sets of well defined and bounded interfaces
  - Functions and Data
- Sets are versioned
  - Well defined lifecycle
- Target specific classes of kernel extensions
  - New classes can be added

The Kext Dependency Problem



Why Apple Needs Kernel Interfaces

- Kexts have no explicit dependencies
- Mac OS X needs to evolve
  - New features
  - Performance
  - Bug fixes
  - New hardware
  - SMP support
- No interfaces—no change
- Interfaces define our “innovation sandbox”

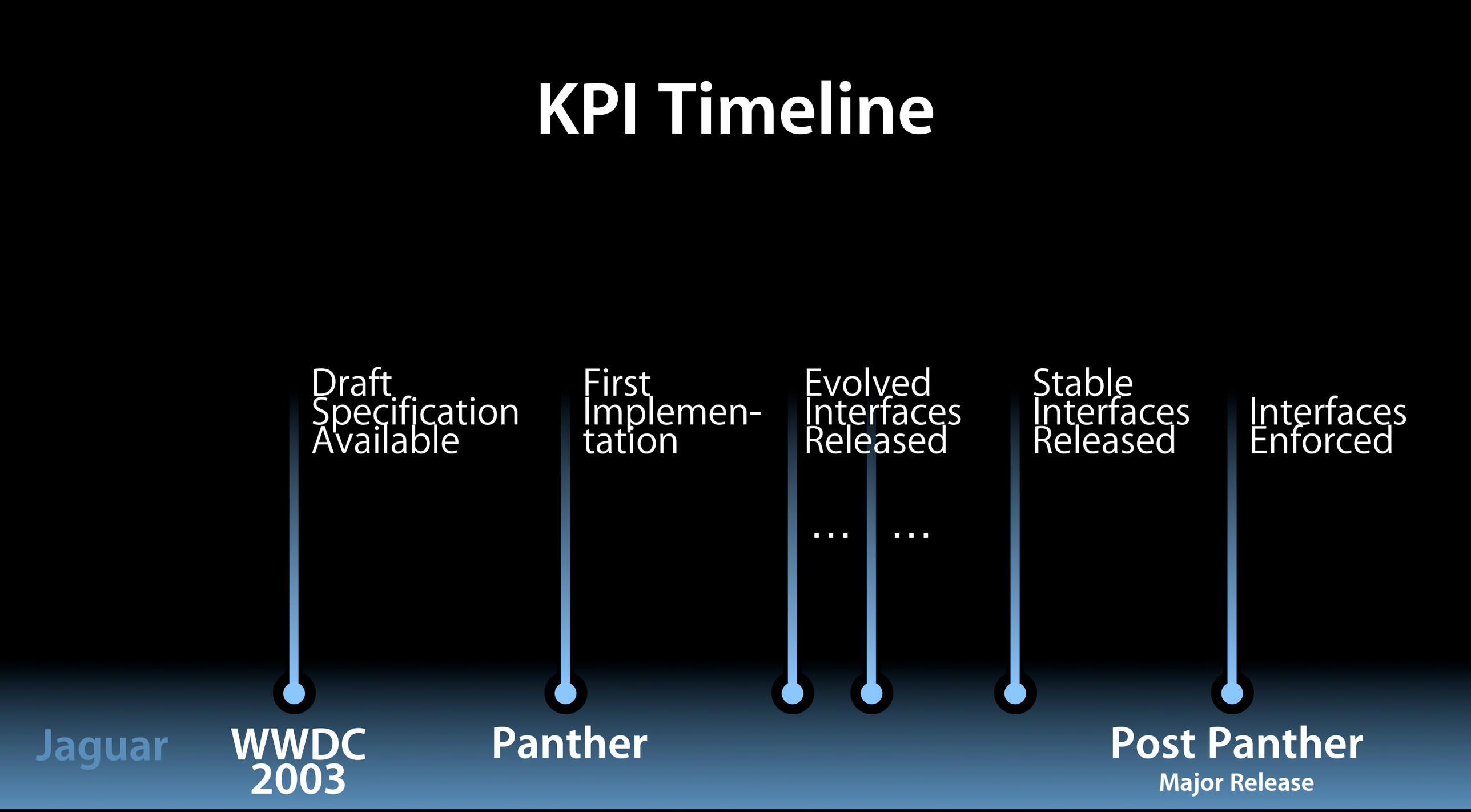
Why Developers Want Kernel Interfaces

- Interfaces best suited for your needs
- Improved stability of environment
- Binary Compatibility from release to release
- Significantly reduce the chance of accidental breakage

Kernel Interface States

Stable	Stable interfaces are guaranteed to be in the next major release
Compatibility	Designed for ease of porting but better interfaces exist. Treated like Stable
Deprecated	Interface is planned to be removed in the following major release
Evolving	Interface is under-development and may change or be removed

KPI Timeline



Kernel Interface Version Management

Dean Reece
Engineering Manager
IOKit Team

KEXT Tools for Panther

- KEXT Tools and linking rules are not changing for Panther
- New capabilities are being added for Panther
- New interface-only KEXT type that will be used for binary compatibility
- KEXTs will no longer link directly against the kernel, but will link against the Interface KEXTs

KEXT Linking In Jaguar

- KEXTs that declare a dependency on any of the following are linked against the whole kernel:
  - com.apple.kernel
  - com.apple.kernel.bsd
  - com.apple.kernel.iokit
  - com.apple.kernel.libkern
  - com.apple.kernel.mach
- True for versions of these dependencies prior to version 7.0

KEXT Linking In Jaguar

com.apple.kernel.* (v1–v6)

com.apple.kernel
6.0

com.apple.kernel.libkern
1.0.0b1–6.0

com.my.kext
com.apple.kernel.bsd: 6.0

com.apple.kernel.bsd
1.1–6.0

com.apple.kernel.mach
1.0.0b1–6.0

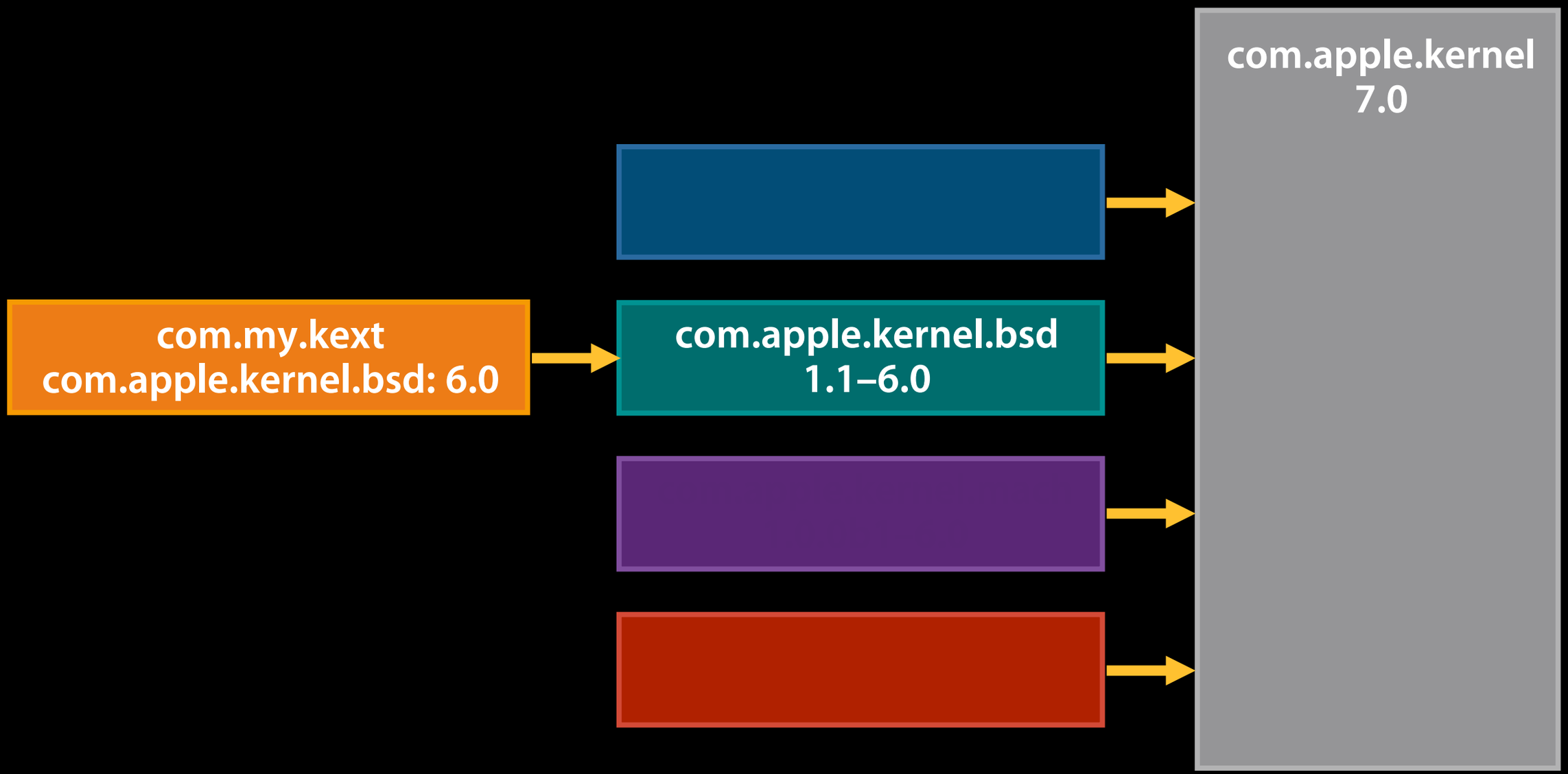
com.apple.kernel.iokit
1.0.0b1–6.0

KEXT Linking In Panther

- In Panther, we will provide binary compatibility with pre-Panther drivers via a set of Interface KEXTs
- These Interface KEXTs will contain (most of) the symbols present in the v6 kernel components
- New Symbols introduced in Panther will not be present in the Interface KEXTs
- The symbols in the Interface KEXTs are simply re-exports of symbols in the real kernel, though they can be implemented under a different name than they are exported

KEXT Linking In Panther

com.apple.kernel.bsd (v1–v6)

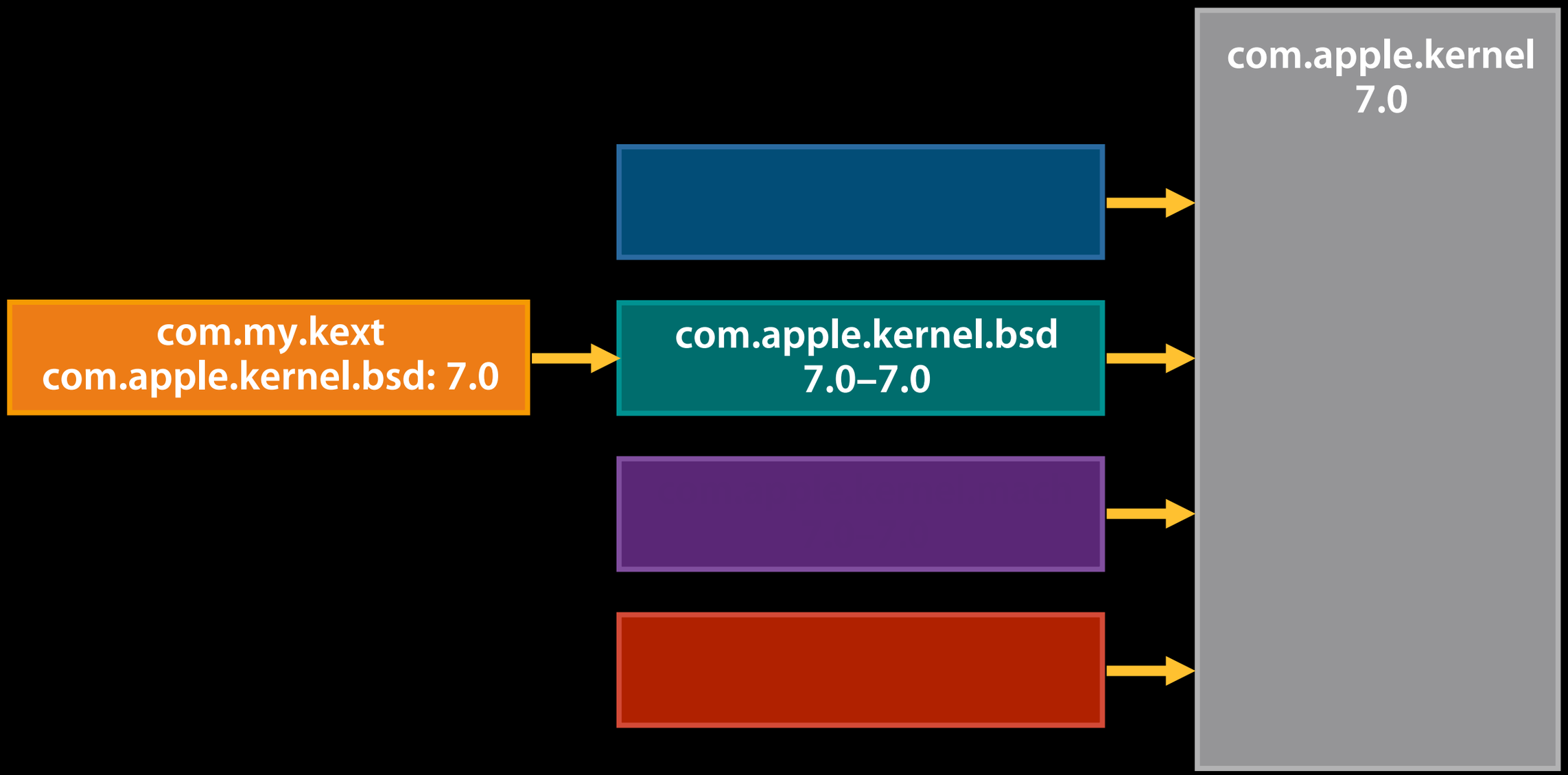


KEXT Linking In Panther

- We will provide a set of Panther-only (v7) Interface KEXTs
- These Interface KEXTs will contain some of the symbols present in the v6 kernel components
- New symbols introduced in Panther will only be available in these v7 Interface KEXTs
- The symbols in the Interface KEXTs are simply re-exports of symbols in the real kernel, though they can be implemented under a different name than they are exported

KEXT Linking In Panther

com.apple.kernel.bsd (v7)

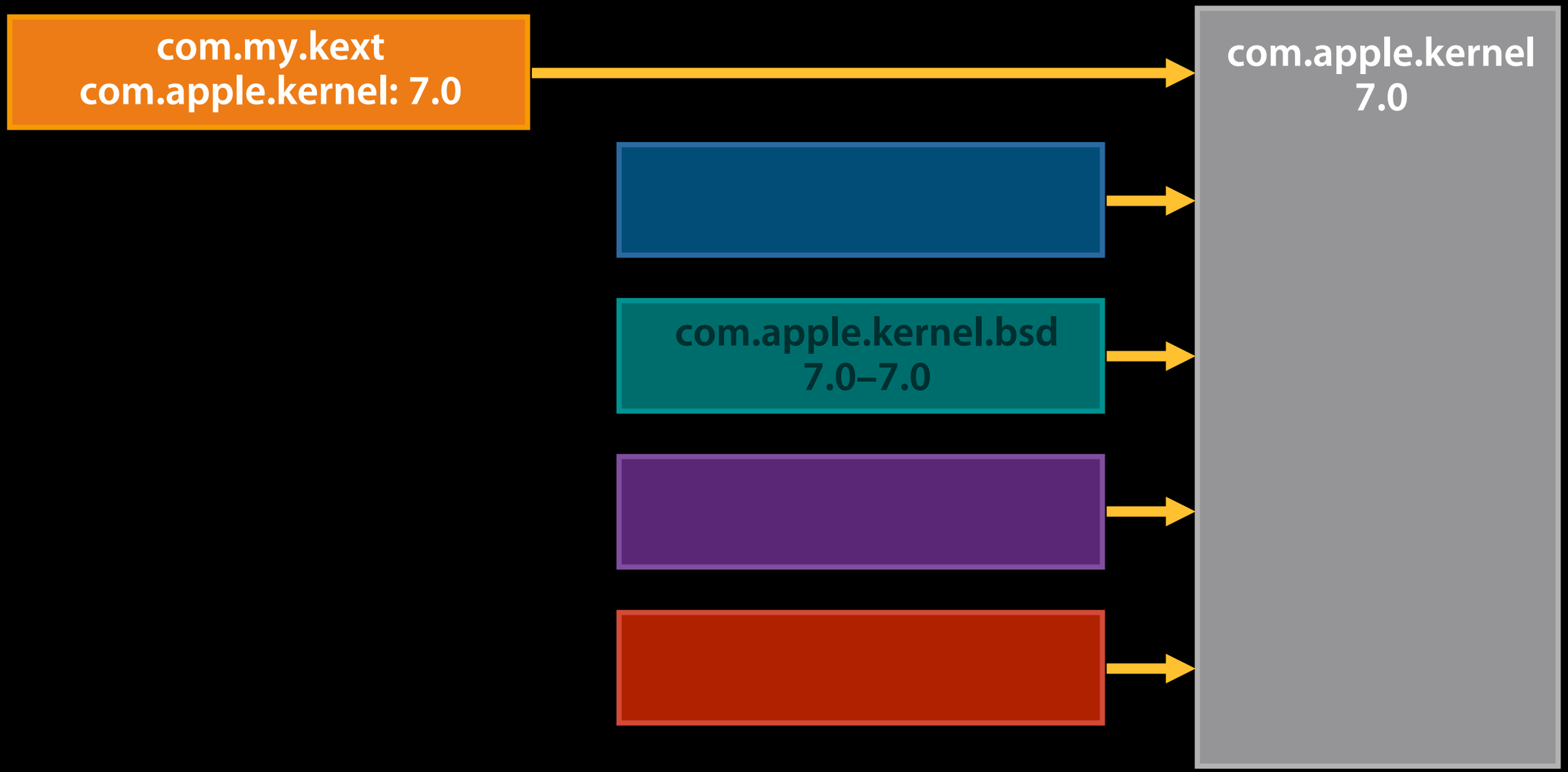


KEXT Linking In Panther

- `com.apple.kernel` continues to represent the whole kernel
- If you specify a dependency on `com.apple.kernel`, your KEXT will link directly against the kernel
- We will continue to update the compatible version with each new release, so no release-to-release binary compatibility will be possible if linking directly against the kernel
- This is primarily useful for development and experimentation (such as at universities) but is not suitable for commercial products

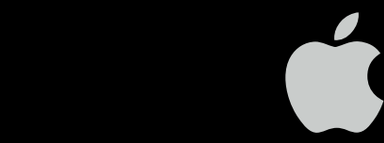
KEXT Linking In Panther

com.apple.kernel (v7)



Interface KEXTs Beyond Panther

- The KEXT system will select the correct Interface KEXT for any `com.apple.kernel` or `com.apple.kpi` dependency
- We will continue to deliver newer Interface KEXTs with each new release of the OS
- We may make Experimental KPIs available to Developers in the form of seed kernels and Interface KEXTs
- As the kernel evolves, we will stop providing the older Interface KEXTs
- This process provides a sliding window of binary compatibility that allows us to evolve KPIs



Kernel Interfaces for I/O Kit

I/O Kit Interfaces In Panther

- The I/O Kit v6 Interface provided in Panther is binary compatible with the Jaguar I/O Kit
- The I/O Kit v7 Interface introduces a few new interfaces
- The I/O Kit v7 Interface removes a few deprecated interfaces

I/O Kit Interfaces New In Panther

- I/O Kit v7 provides extensions to the `IOMemoryDescriptor` classes for large memory addressing
- I/O Kit v7 provides new functions for easy access to physical memory:

```
IOMappedRead8()    IOMappedWrite8()
IOMappedRead16()   IOMappedWrite16()
IOMappedRead32()   IOMappedWrite32()
IOMappedRead64()   IOMappedWrite64()
```

I/O Kit Interfaces Removed From Panther

- I/O Kit v7 obsoletes the following classes:
  - `IOCommandQueue`
  - `IOSyncer`

General and Filesystem Interfaces

Mark Gorlinsky
Engineering Manager
Kernel Team

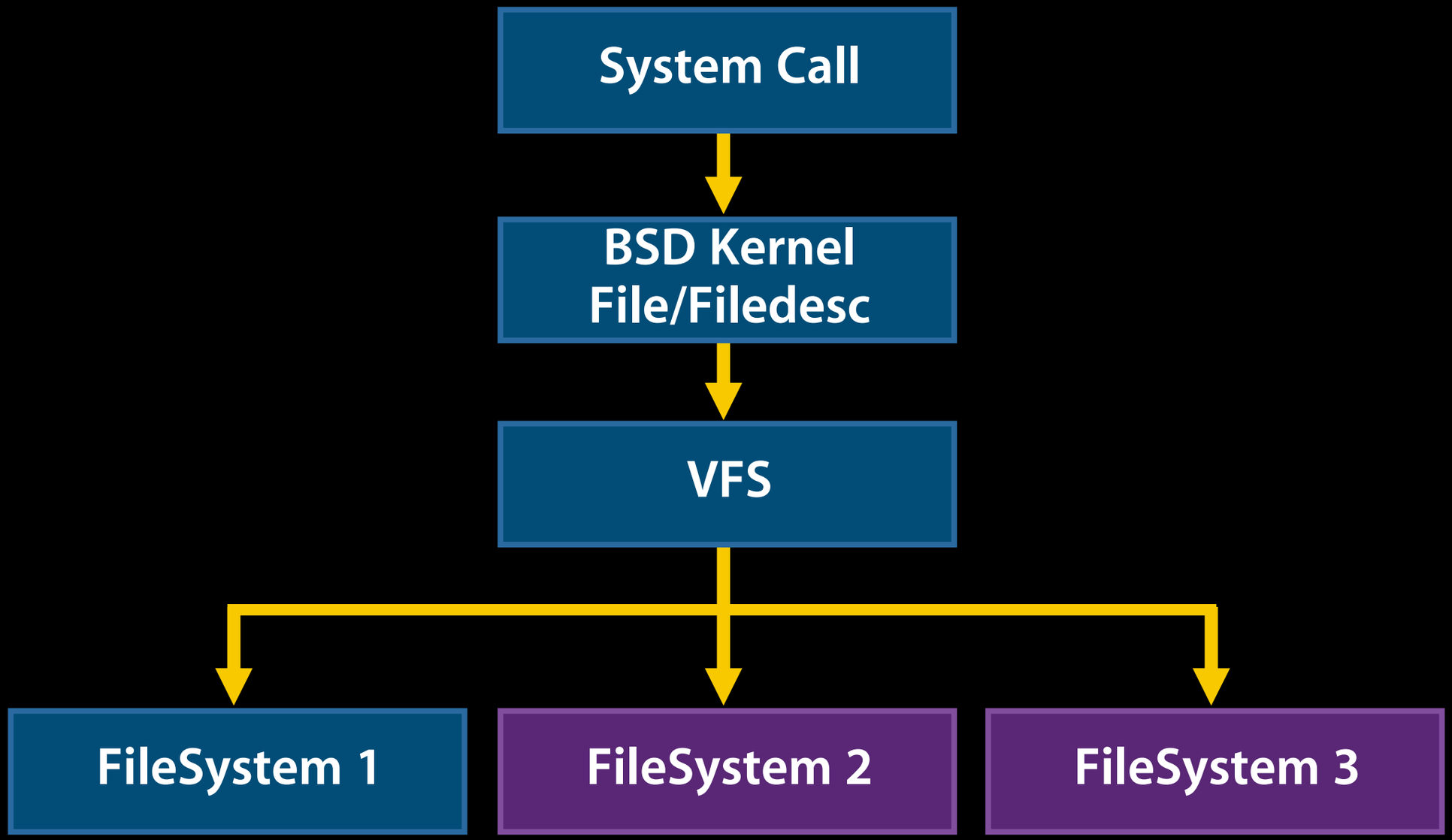
Moving Forward

- SMP support
  - Funnels -> thread safe with fine grain locking
- Data structures require extensive changes
  - Global access -> access through interfaces
  - Adding reference and usage counts
- Lock exposure
- New MP safe interfaces

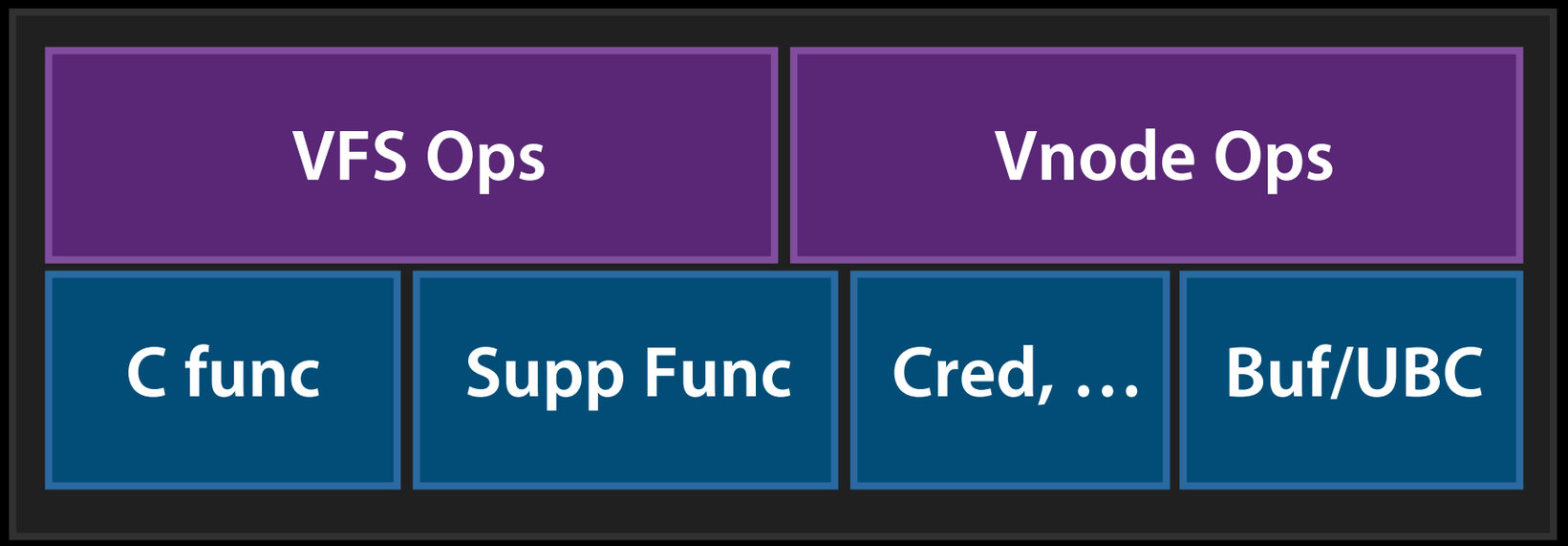
Other Issues to be Addressed

- 32-bit block numbers -> 64-bit
- Difficult to create stackable file systems
- Move common functionality to generic VFS layer
- Cannot evolve or address issues that break binary compatibility

Kernel Interfaces Overview

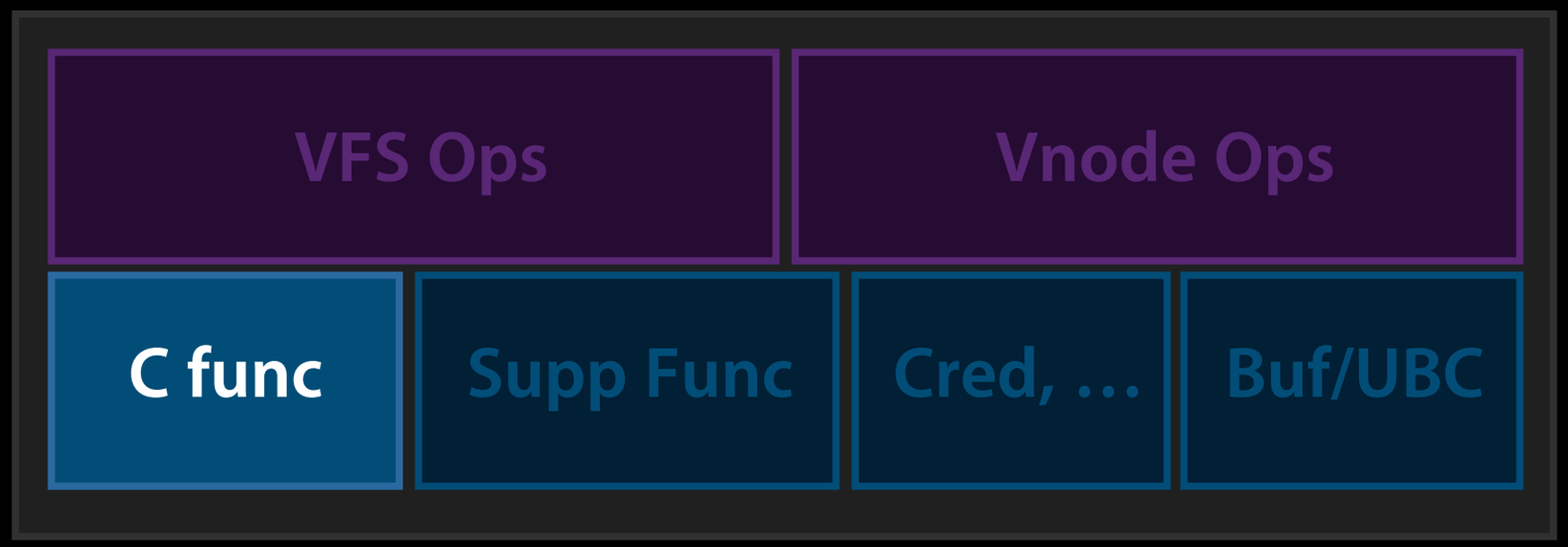


File System Overview



File System Overview

C Library Functions



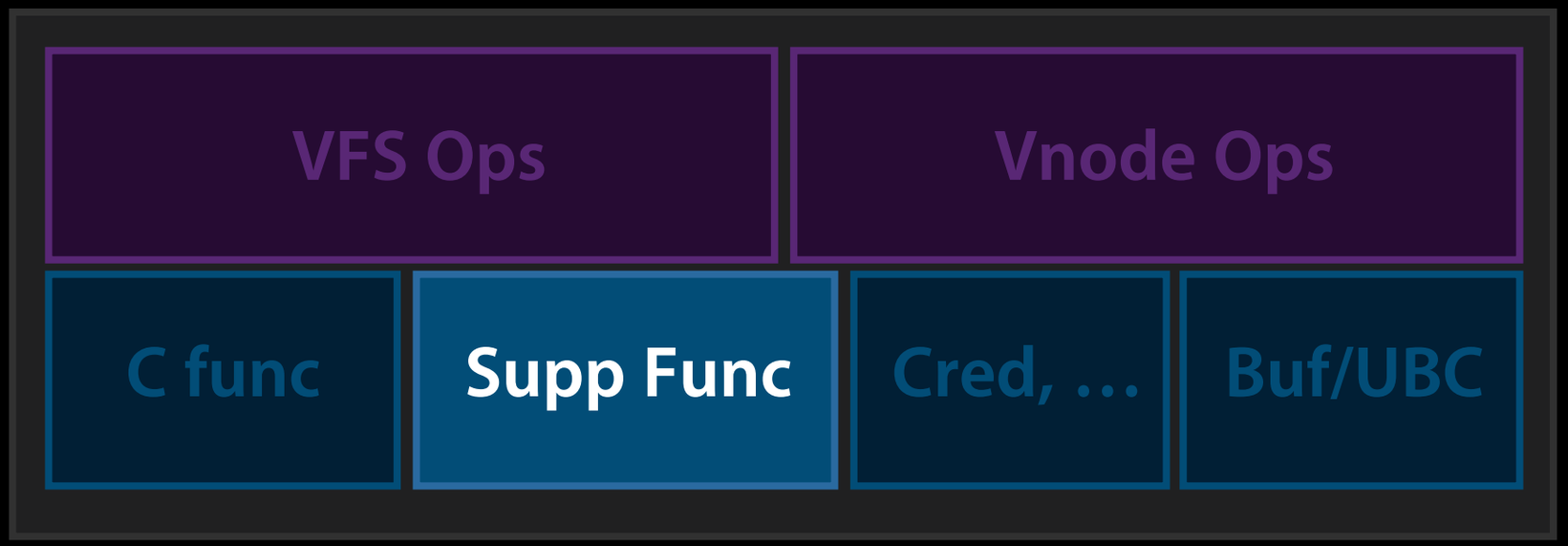
C Library Functions

Sub-set of commonly used C functions

- String manipulation
  - strcat, strcpy, …
- Memory manipluation
  - bcopy, bcmp, …
- Misc functions
  - abs, atoi, ffs, …

File System Overview

Kernel Support Interfaces



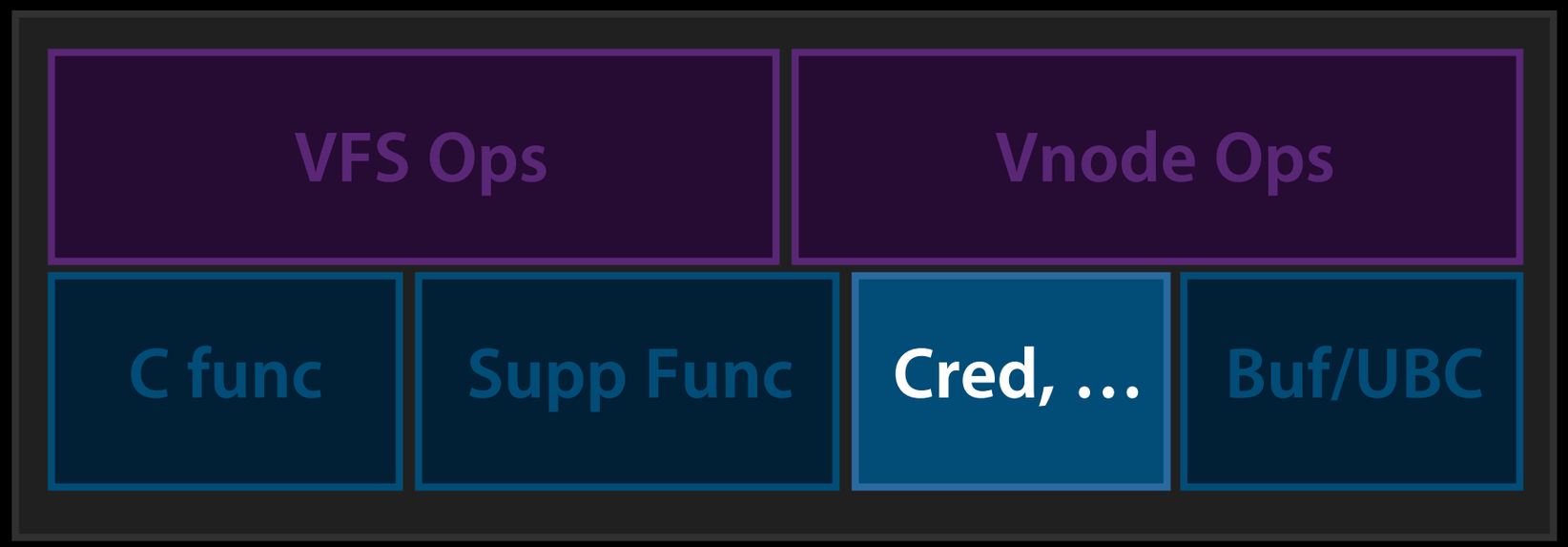
Kernel Support Interfaces

Basic functions for resource management and thread control

- Memory management
- Synchronization
- Thread control
- User/kernel data copy functions

File System Overview

BSD Kernel Interfaces

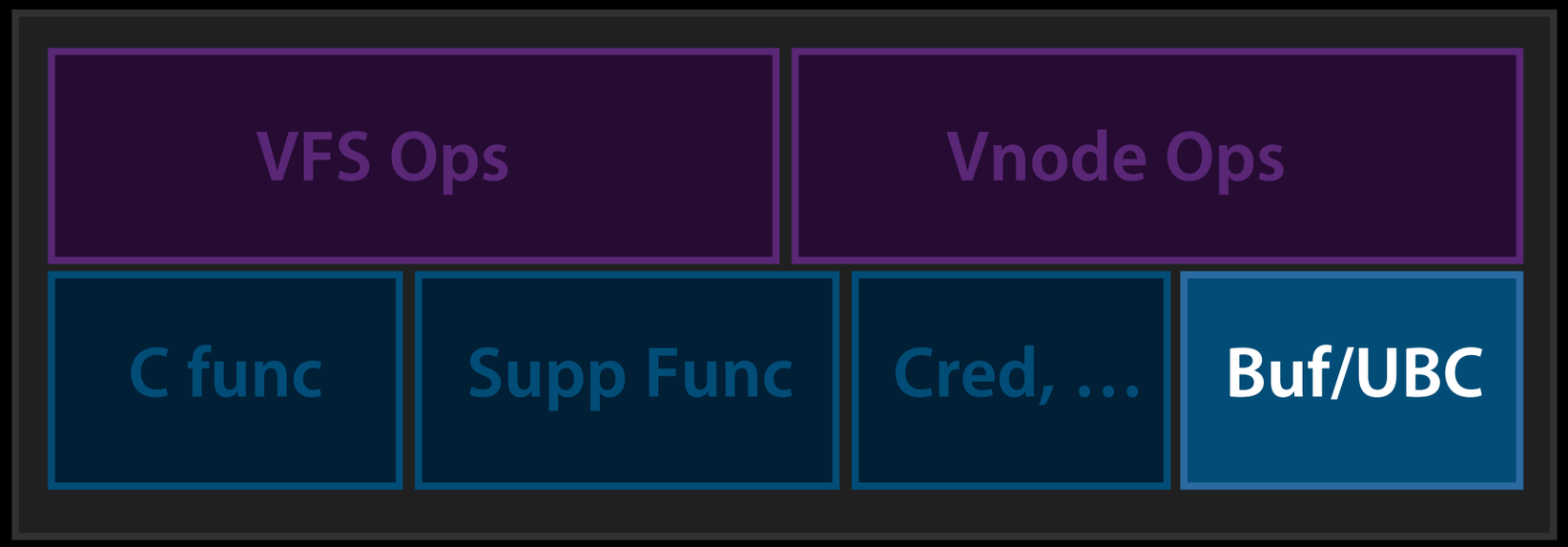


BSD Kernel Interfaces

- Process credentials, id, limits
- File descriptor to vnode or socket
- Addition of other areas based on need
- Some interfaces cannot be exposed
  - sysent

File System Overview

File Caching



File Caching

- Buffer cache
- Semantics are same except 64-bit block numbers
- UBC
- Remain largely unchanged
  - Macros are now functions

File System Overview

VFS Interfaces

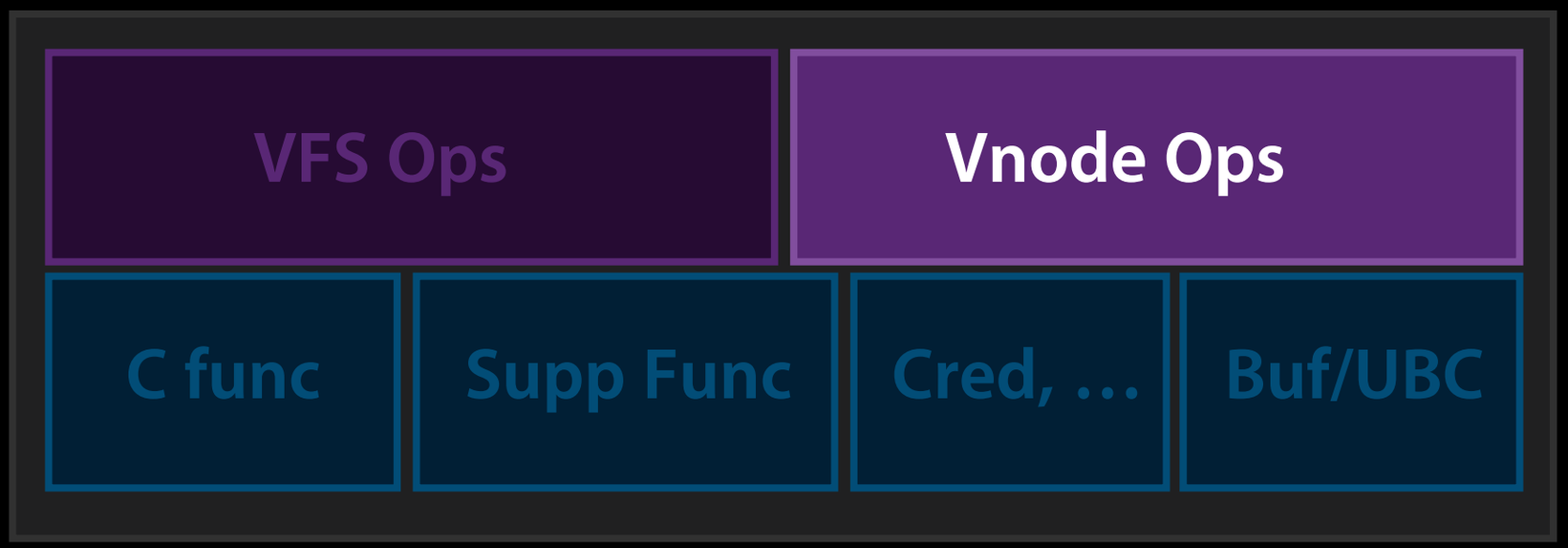
VFS Interfaces

Interfaces for volume manipulation

- “mount_t” data type
- New statfs and IO attributes structure
- New VFS OPs
- Extensible vfsops

File System Overview

Vnode Interfaces



Vnode Interfaces

Interfaces for file manipulation

- “vnode_t” data type
- Symmetric vnode ops
- New vnode create call
- vnode suspend and resume
- vnode_vattr instead vattr

Network Interfaces

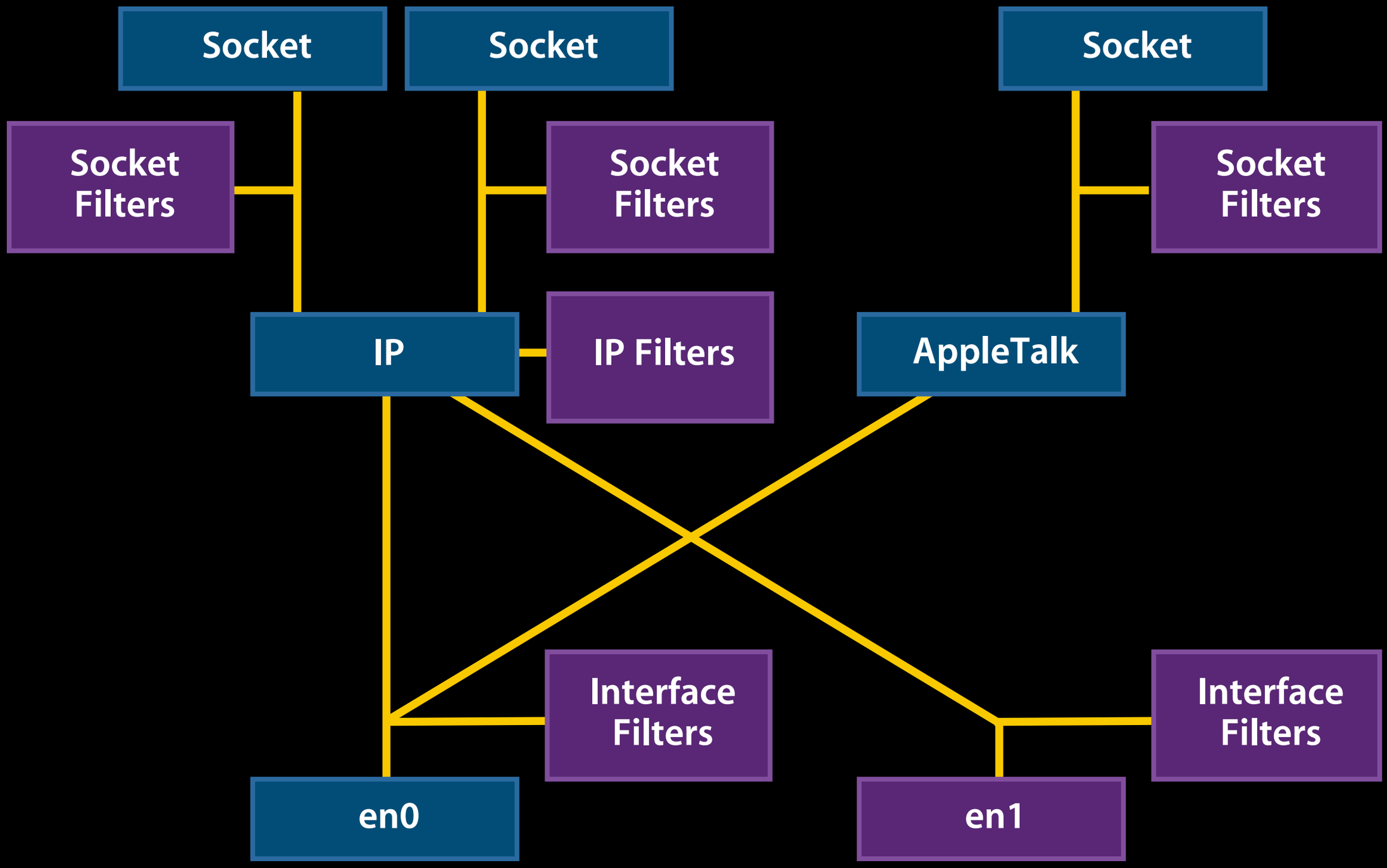
Josh Graessley
Senior Engineer
Networking Team

Network Kernel Interfaces

- mbuf
- Socket
- Socket Filter
- IP Filter
- Interface Filter
- Interface

Network Stack Overview

Network Stack



Network Kernel Interfaces Overview

- All kernel internal data structures are opaque
- Filters based on operations rather than implementation
- Functions return errors when appropriate
- Additional future interface sets being investigated
  - Support for new interface families
  - Support for new protocols

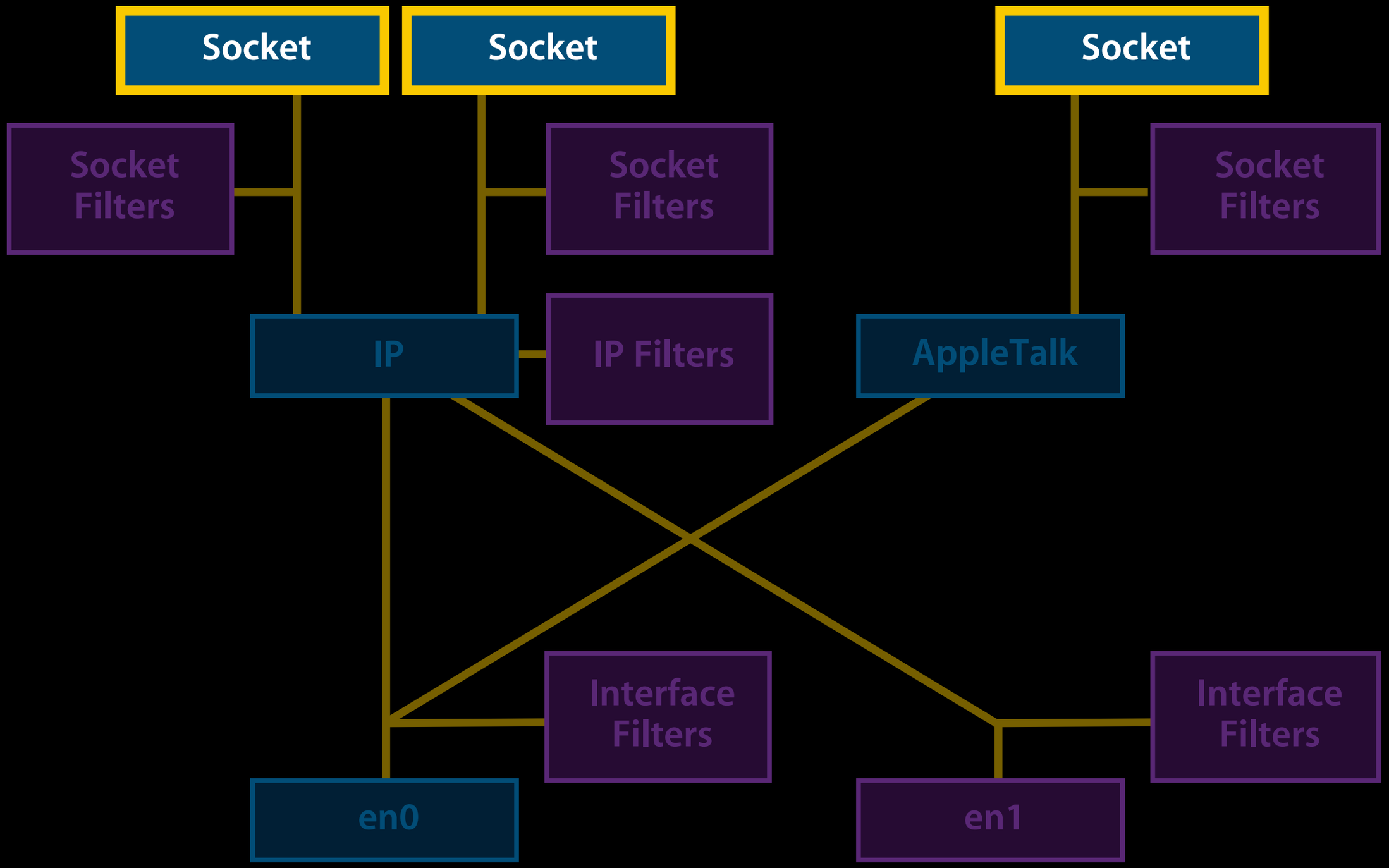
Mbuf KPI

Access to data in mbufs

- Opaque data structure `mbuf_t`
- Accessor functions to get/set data in the buffer
- Consistent naming and usage with *BSD
  - `mtod()` becomes `mbuf_mtod()`
  - `m_freem()` becomes `mbuf_freem()`
- Most functions can return errors

Network Stack Overview

Socket



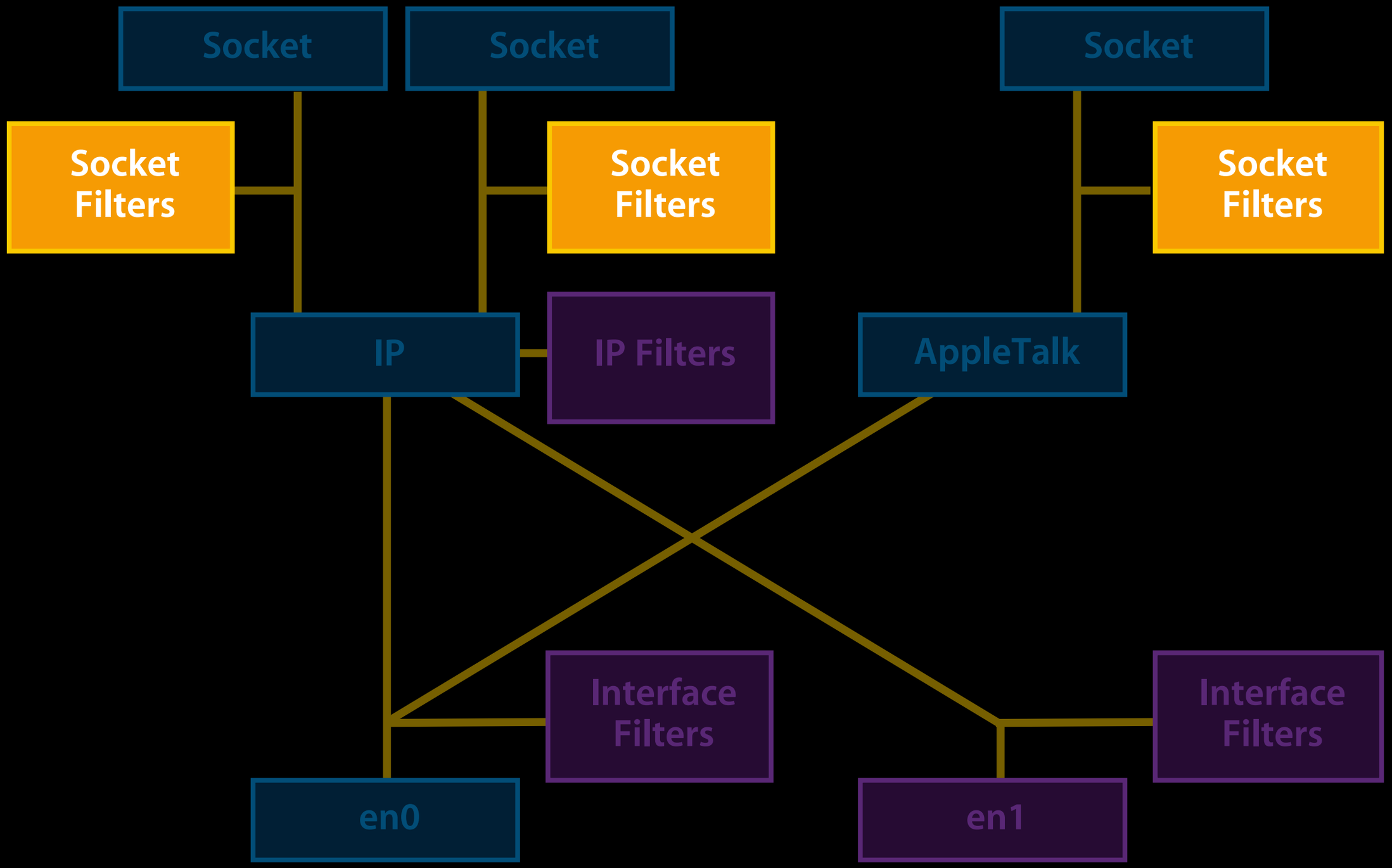
Socket KPI

Make sockets available in the kernel e.g., for network filesystems

- Modeled after userspace APIs
  - `sock_socket()`, `sock_bind()`
  - Socket and protosw structures opaque
    - No direct access
  - Hide Protocol control block
    - use `sock_getsockname()` `sock_getpeername()`, and `sock_getsockopt()`

Network Stack Overview

Socket Filter



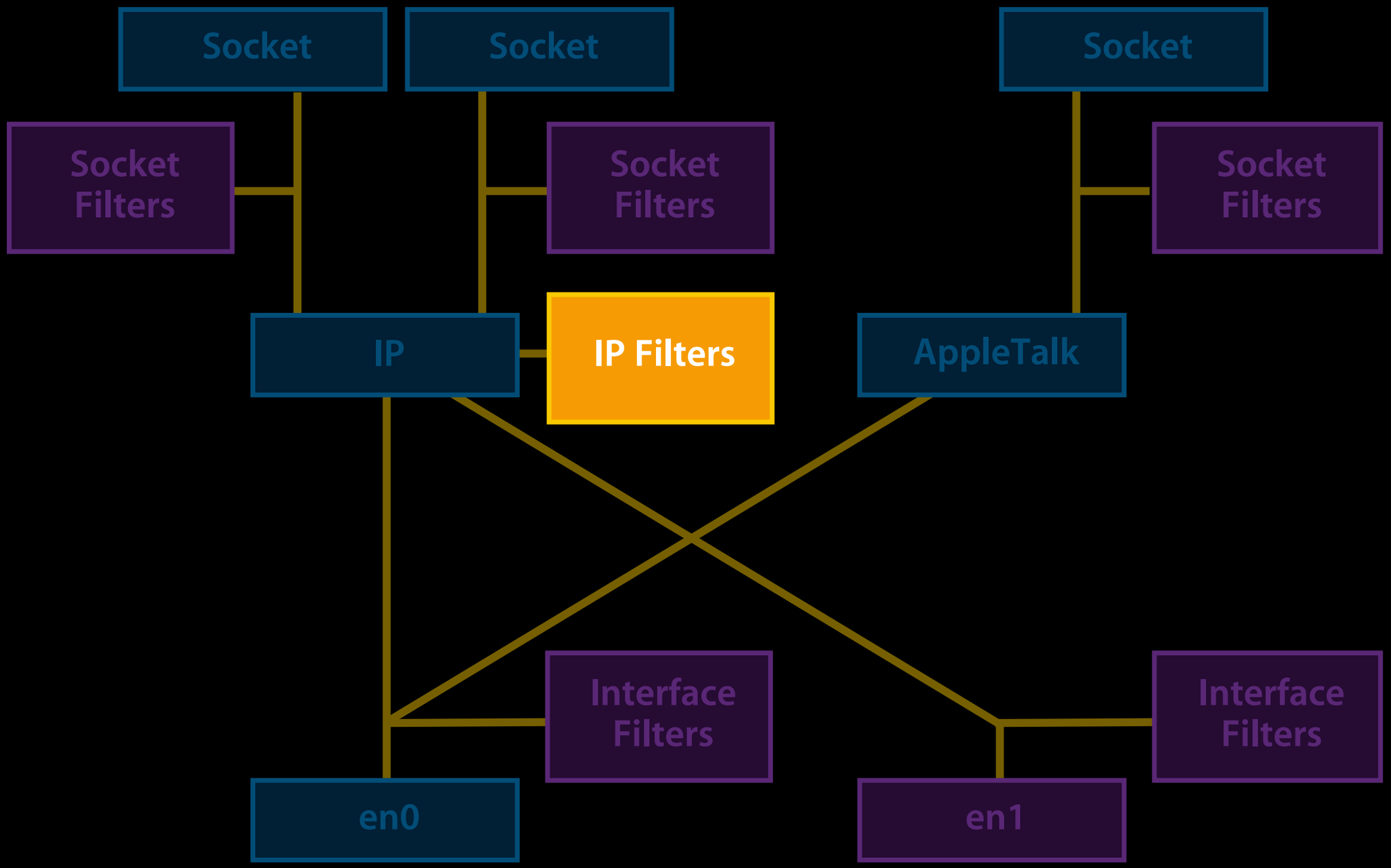
Socket Filter KPI

Filter data and commands at the socket layer

- Rewritten socket filters
- Receive notification of socket state changes
- Filter in/outbound data
- Filter in/outbound connections
- Filter binds
- Filter get/setsockopt
- Filter listen
- Filter ioctl

Network Stack Overview

IP Filter



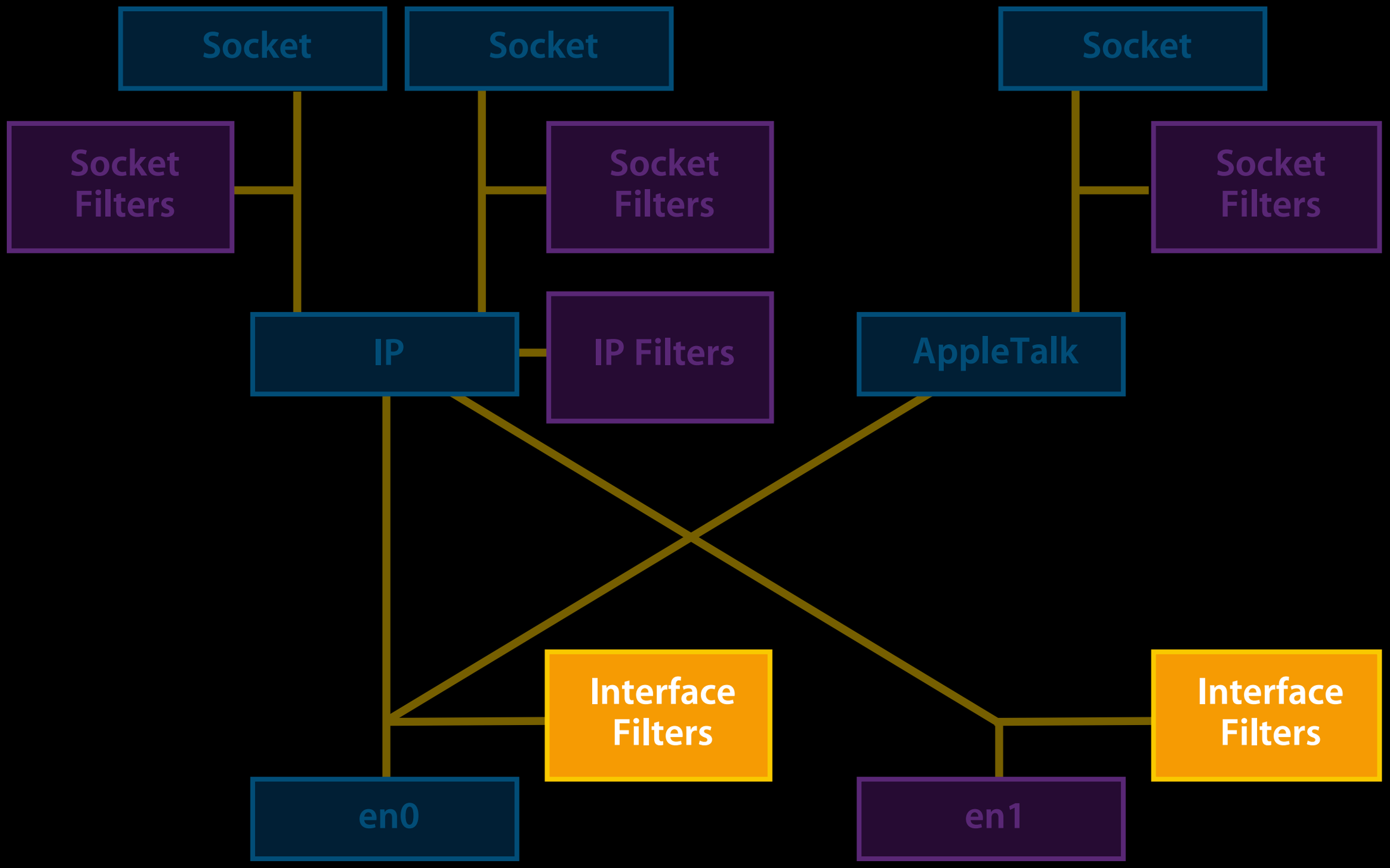
IP Filter KPI

Filter packets at the IP layer for content/firewall/etc.

- New place to hook in
- Filter in/outbound IP packets
- Filter inbound after packet reassembly
- Filter inbound before and after IPSec, tunneling, etc.
- Filter outbound before disassembly
- Filter outbound before IPSec
- Filter IPv4 and/or IPv6

Network Stack Overview

Network Interface Filter



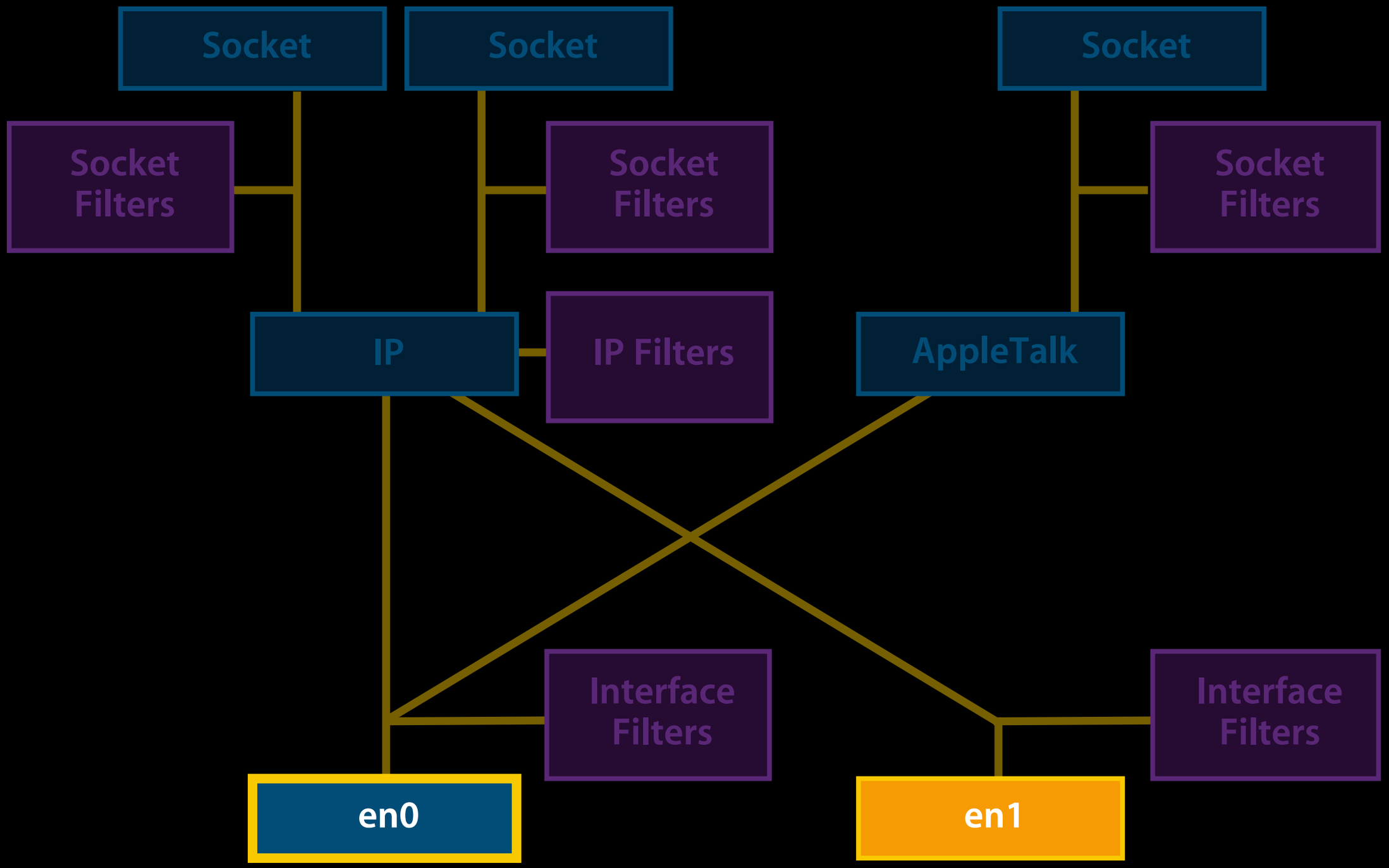
Network Interface Filter KPI

Filter packets on an interface

- Based on DLIL interface filters
- Filter inbound/outbound packets on an interface
- Filter ioctls on an interface
- Filter events on an interface

Network Stack Overview

Network Interface



Network Interface KPI

Provides access to network interfaces (ifnets)

- Interfaces are opaque and reference counted
- Accessors for inspecting
- Retrieve list of interfaces
- Retrieve list of addresses
- Join/leave multicasts
- Add an interface
  - Must be to an existing interface family (e.g., ethernet, ppp, firewire)

Summary

Simon Patience
Director of Engineering
Core OS

To Summarize—Kernel Interfaces

- Allow Apple to innovate in the kernel
- Allow Apple to improve the quality of the kernel
- Allow Developers to write KEXTs more easily
- Future proofing for Developers of KEXTs
- Provide a predictable rate of change for Developers
- A better user experience when KEXTs get out of date
- Start today!

Call to Arms

- Download the draft Kernel Interface specifications
- Review the interfaces against your KEXTs
- Send us feedback
  - kpi-feedback@apple.com
- Track the updates
- Port your KEXTs to use the interfaces
- Work with Apple to make this a success for everyone

Wrap Up

Roadmap

100 IOKit: Kernel Extension Programming Techniques for Writing Kernel Extensions	**Russian Hill** **Mon., 3:30pm**
102 Open Source at Apple An Overview of Apple's Open Source Efforts	**The Mission** **Tues., 10:30am**
501 Bluetooth Update Bluetooth in Mac OS X	**The Marina** **Tues., 9:00am**
503 USB Update USB in Mac OS X	**The Marina** **Tues., 2:00pm**
505 Firewire in Depth	**The Marina** **Tues., 5:00pm**
FF001 Darwin Tell Us What You Think	**North Beach** **Tues., 3:30pm**
FF006 Firewire and USB Tell Us What You Think	**North Beach** **Wed., 3:30pm**

Who to Contact

Craig Keithley
I/O Technology Evangelist
keithley@apple.com

http://developer.apple.com/wwdc2003/urls.html

For More Information

Reference Library

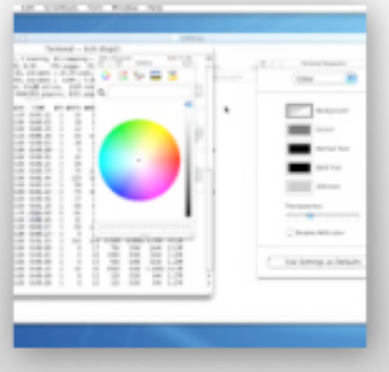
Documentation

Inside Mac OS X: Kernel Programming
ADC Home > Documentation > Darwin & Core OS >

I/O Kit Fundamentals
ADC Home > Documentation > Device Drivers & I/O Kit >

Device Drivers (Kernel-I/O Kit) API Reference
ADC Home > Documentation > Device Drivers & I/O Kit > Device Drivers (Kernel-I/O Kit)

Kernel Extensions (Kernel/libkern) API Reference
ADC Home > Documentation > Device Drivers & I/O Kit > Kernel Extensions (Kernel/libkern)



Q&A

Craig Keithley
I/O Technology Evangelist
keithley@apple.com

http://developer.apple.com/wwdc2003/urls.html

WWDC 2003

Worldwide Developers Conference